GitHub使用

1. 初始化一个Git仓库，使用git init命令。
2. 创建一个版本库非常简单，首先，选择一个合适的地方，创建一个空目录：
3. $ mkdir learngit
4. $ cd learngit
5. $ pwd
6. C:/Users/lee/learngit
7. pwd命令用于显示当前目录。在我的电脑上，这个仓库位于C:/Users/lee/learngit
8. 第二步，通过git init命令把这个目录变成Git可以管理的仓库：
9. $ git init
10. 瞬间Git就把仓库建好了，而且告诉你是一个空的仓库（empty Git repository），细心的读者可以发现当前目录下多了一个.git的目录（隐藏文件):
11. 这个目录是Git来跟踪管理版本库的，没事千万不要手动修改这个目录里面的文件，不然改乱了，就把Git仓库给破坏了。
12. git status

查看本地修改后的工程与上一版工程之间的更改状态信息；

1. git diff “文件名”

虽然Git告诉我们readme.txt被修改了，但如果能看看具体修改了什么内容，自然是很好的。比如你休假两周从国外回来，第一天上班时，已经记不清上次怎么修改的readme.txt，所以，需要用这个命令。

1. git add code\_file\_name1 code\_file\_name2 … code\_file\_namen

添加的代码文件到git仓库

1. git commit -m "任意起名"

提交更改后的工程到git仓库（git commit 主要是将暂存区里的改动给提交到本地的版本库。）

1. git push

在GitHub账户上更新工程

1. 首先，登陆GitHub，然后，在右上角找到“Create a new repo”按钮，创建一个新的仓库：

在Repository name填入learngit，其他保持默认设置，点击“Create repository”按钮，就成功地创建了一个新的Git仓库：

目前，在GitHub上的这个learngit仓库还是空的，GitHub告诉我们，可以从这个仓库克隆出新的仓库，也可以把一个已有的本地仓库与之关联;

然后，把本地仓库的内容推送到GitHub仓库。

现在，我们根据GitHub的提示，在本地的learngit仓库下运行命令：

$ git remote add origin https://github.com/starlee261/gaonaoshi.git

请千万注意，把上面的starlee261替换成你自己的GitHub账户名，否则，你在本地关联的就是我的远程库;

关联没有问题，但是你以后推送是推不上去的，因为你的SSH Key公钥不在我的账户列表中。

添加后，远程库的名字就是origin，这是Git默认的叫法，也可以改成别的，但是origin这个名字一看就知道是远程库。

1. 下一步，就可以把本地库的所有内容推送到远程库上：

$ git push -u origin master

把本地库的内容推送到远程，用git push命令，实际上是把当前分支master推送到远程。

由于远程库是空的，我们第一次推送master分支时，加上了-u参数;

Git不但会把本地的master分支内容推送的远程新的master分支，还会把本地的master分支和远程的master分支关联起来

1. 在以后的推送或者拉取时就可以简化命令。

推送成功后，可以立刻在GitHub页面中看到远程库的内容已经和本地一模一样：

从现在起，只要本地作了提交，就可以通过命令：

$ git push origin master

把本地master分支的最新修改推送至GitHub，现在，你就拥有了真正的分布式版本库！

1. git reset - -hard commit\_id 时光机穿梭

HEAD指向的版本就是当前版本，因此，Git允许我们在版本的历史之间穿梭，使用命令git reset - -hard commit\_id。 在Git中，用HEAD表示当前版本，，上一个版本就是HEAD^，上上一个版本就是HEAD^^，当然往上100个版本写100个^比较容易数不过来，所以写成HEAD~100

1. 穿梭前，用git log可以查看提交历史，以便确定要回退到哪个版本。
2. 要重返未来，用git reflog查看命令历史，以便确定要回到未来的哪个版本。
3. 提交后，用git diff HEAD -- readme.txt命令可以查看工作区和版本库里面最新版本的区别：

未完待续：

2019.05.23

操作记录如下：